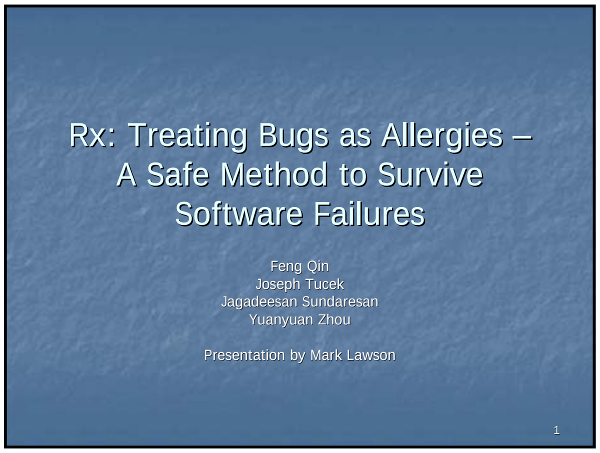
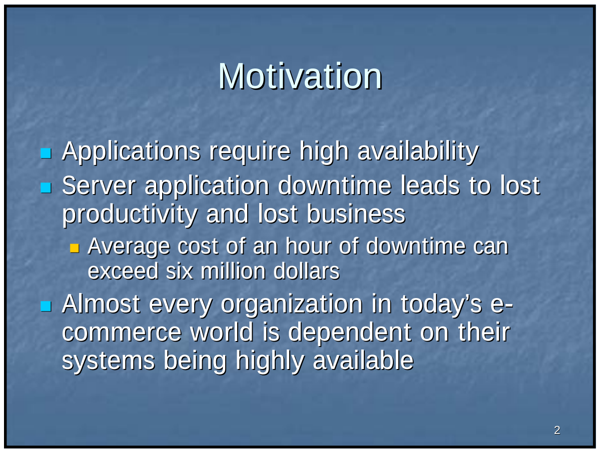


# Rx: Treating Bugs as Allergies – A Safe Method to Survive Software Failures

Feng Qin
Joseph Tucek
Jagadeesan Sundaresan
Yuanyuan Zhou

Presentation by Mark Lawson

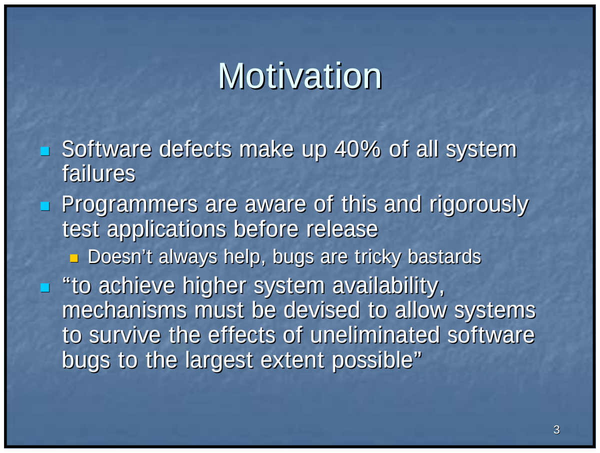
1



## Motivation

- Applications require high availability
- Server application downtime leads to lost productivity and lost business
  - Average cost of an hour of downtime can exceed six million dollars
- Almost every organization in today's e-commerce world is dependent on their systems being highly available

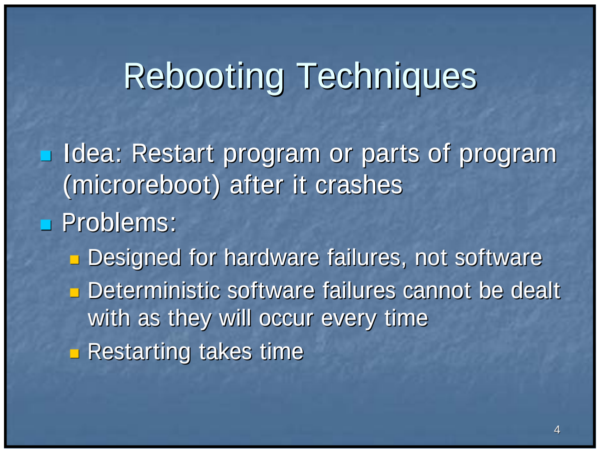
2



## Motivation

- Software defects make up 40% of all system failures
- Programmers are aware of this and rigorously test applications before release
  - Doesn't always help, bugs are tricky bastards
- "to achieve higher system availability, mechanisms must be devised to allow systems to survive the effects of uneliminated software bugs to the largest extent possible"

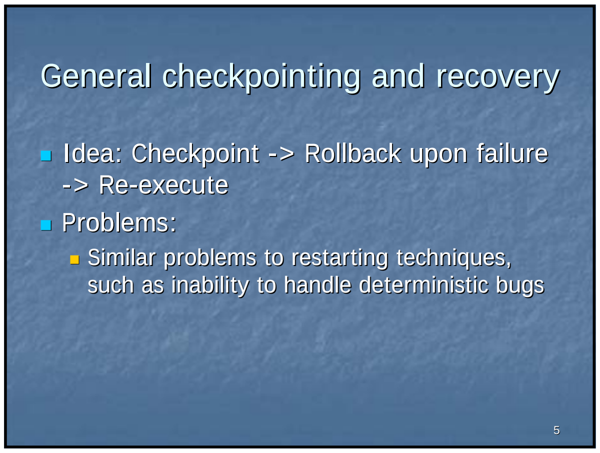
3



## Rebooting Techniques

- Idea: Restart program or parts of program (microreboot) after it crashes
- Problems:
  - Designed for hardware failures, not software
  - Deterministic software failures cannot be dealt with as they will occur every time
  - Restarting takes time

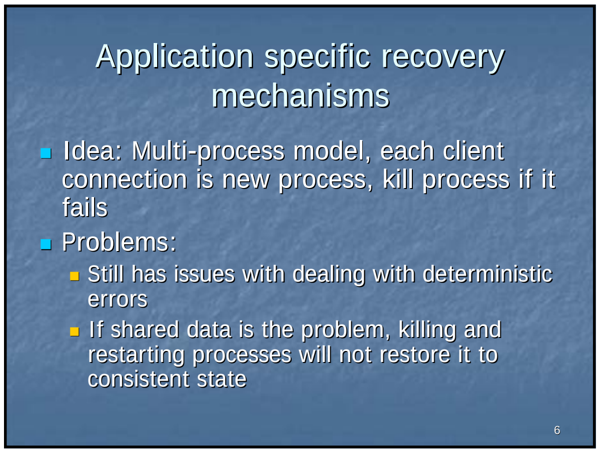
4



## General checkpointing and recovery

- Idea: Checkpoint -> Rollback upon failure -> Re-execute
- Problems:
  - Similar problems to restarting techniques, such as inability to handle deterministic bugs

5



## Application specific recovery mechanisms

- Idea: Multi-process model, each client connection is new process, kill process if it fails
- Problems:
  - Still has issues with dealing with deterministic errors
  - If shared data is the problem, killing and restarting processes will not restore it to consistent state

6

## Other methods

- Failure-oblivious computing
  - Idea: Provide artificial values for out-of-bound reads
- Reactive immune system
  - Idea: Creates emulators to run “faulty” regions of a program
- Problems:
  - Considered by authors as “unsafe” because they mask behaviors and speculate as to what the program wants to achieve
  - Immune system has large overheads

7

## Rx real-world metaphor

- Idea: Treat software bugs as real-world allergies
- In real life allergens can be dealt with by changing living environment
  - Removing cat hair from area allows me to breathe better
- Successfully removing allergen from environment allows one to determine cause of allergy
  - No cat hair = no sneezing → allergic to cats

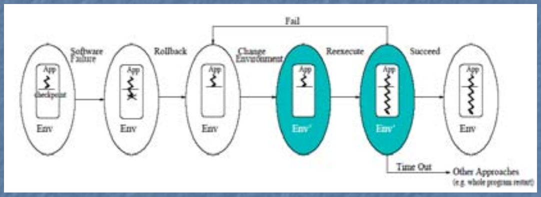
8

## Rx metaphor implemented

- Bugs resemble allergies
- Bugs can be dealt with by changing execution environment
- When a bug is detected, rollback to checkpoint and alter execution environment to deal with detected issues
- Least-intrusive changes can be tried first and more drastic changes can be implemented until a good execution environment is found

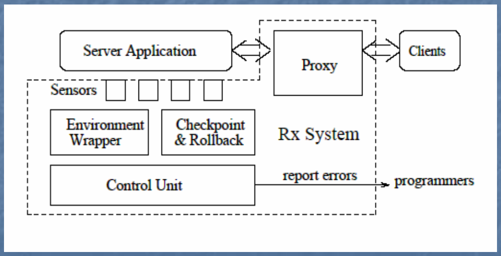
9

## The Main Idea



10

## Rx Architecture

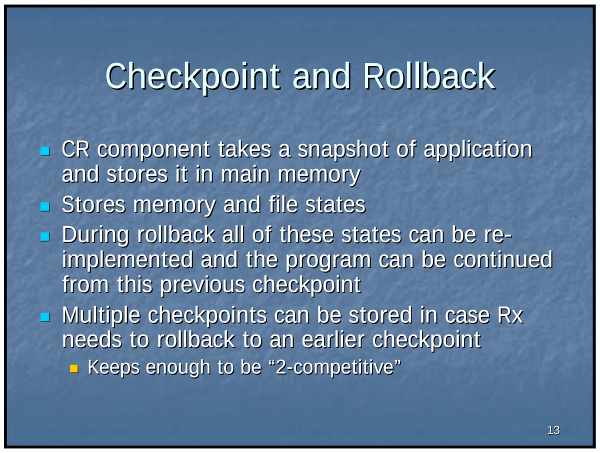


11

## Sensors

- Dynamically monitor applications execution to determine software failures
- Sends information to control unit
- Two types of sensors
  - Sensor to monitor software errors (assertion failures, access violations)
  - Sensor to monitor software bugs (buffer overflows, access to freed memory)

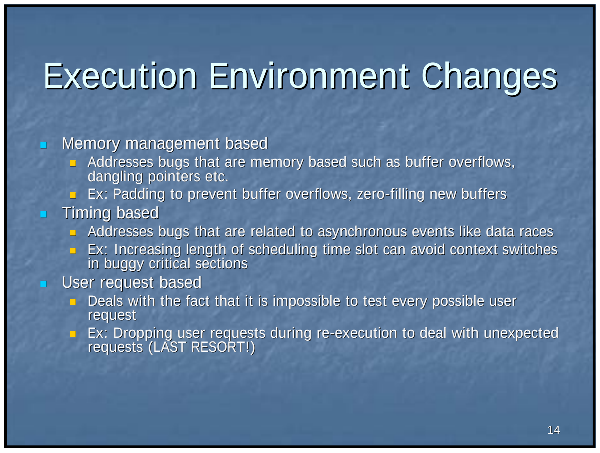
12



## Checkpoint and Rollback

- CR component takes a snapshot of application and stores it in main memory
- Stores memory and file states
- During rollback all of these states can be re-implemented and the program can be continued from this previous checkpoint
- Multiple checkpoints can be stored in case Rx needs to rollback to an earlier checkpoint
  - Keeps enough to be "2-competitive"

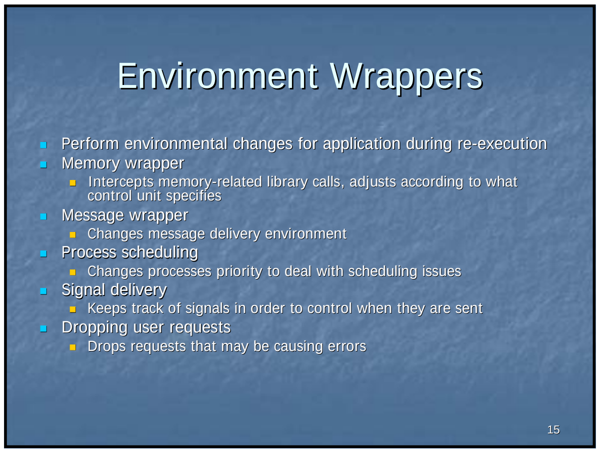
13



## Execution Environment Changes

- Memory management based
  - Addresses bugs that are memory based such as buffer overflows, dangling pointers etc.
  - Ex: Padding to prevent buffer overflows, zero-filling new buffers
- Timing based
  - Addresses bugs that are related to asynchronous events like data races
  - Ex: Increasing length of scheduling time slot can avoid context switches in buggy critical sections
- User request based
  - Deals with the fact that it is impossible to test every possible user request
  - Ex: Dropping user requests during re-execution to deal with unexpected requests (LAST RESORT!)

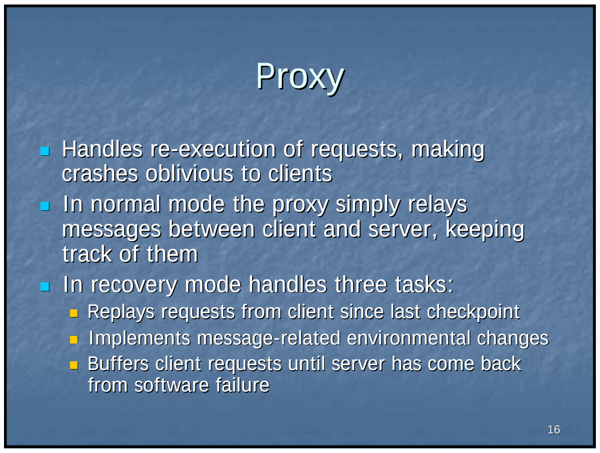
14



## Environment Wrappers

- Perform environmental changes for application during re-execution
- Memory wrapper
  - Intercepts memory-related library calls, adjusts according to what control unit specifies
- Message wrapper
  - Changes message delivery environment
- Process scheduling
  - Changes processes priority to deal with scheduling issues
- Signal delivery
  - Keeps track of signals in order to control when they are sent
- Dropping user requests
  - Drops requests that may be causing errors

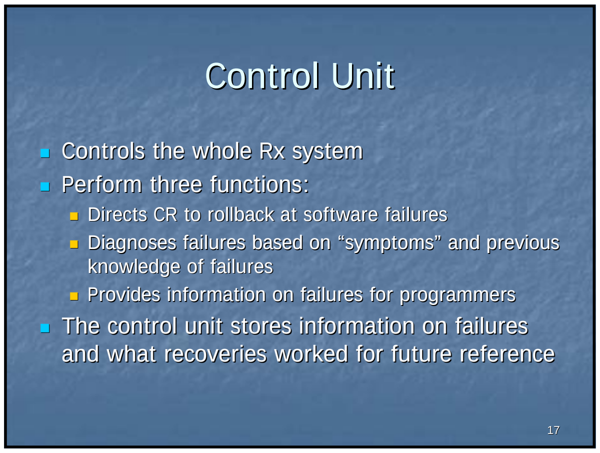
15



## Proxy

- Handles re-execution of requests, making crashes oblivious to clients
- In normal mode the proxy simply relays messages between client and server, keeping track of them
- In recovery mode handles three tasks:
  - Replays requests from client since last checkpoint
  - Implements message-related environmental changes
  - Buffers client requests until server has come back from software failure

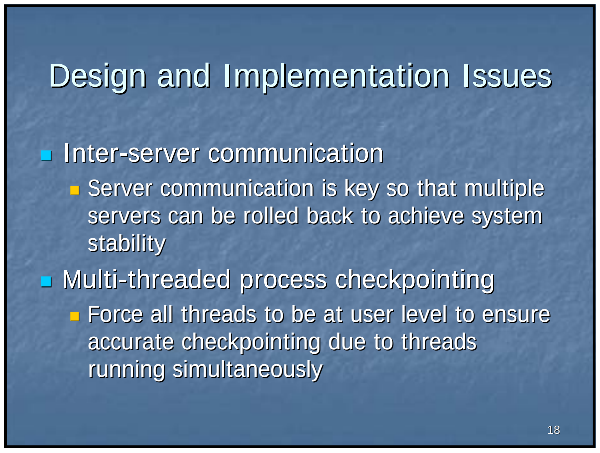
16



## Control Unit

- Controls the whole Rx system
- Perform three functions:
  - Directs CR to rollback at software failures
  - Diagnoses failures based on "symptoms" and previous knowledge of failures
  - Provides information on failures for programmers
- The control unit stores information on failures and what recoveries worked for future reference

17



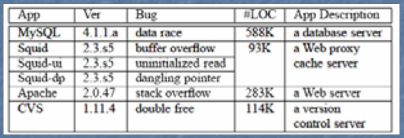
## Design and Implementation Issues

- Inter-server communication
  - Server communication is key so that multiple servers can be rolled back to achieve system stability
- Multi-threaded process checkpointing
  - Force all threads to be at user level to ensure accurate checkpointing due to threads running simultaneously

18

## Evaluation

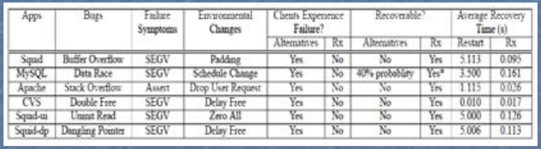
- Tested on 4 server applications (Apache httpd, MySQL, Squid, CVS)



| App | Ver | Bug | #LOC | App Description |
|---|---|---|---|---|
| MySQL | 4.1.1.a | data race | 588K | a database server |
| Squid | 2.3.s5 | buffer overflow | 93K | a Web proxy cache server |
| Squid-ui | 2.3.s5 | uninitialized read | | |
| Squid-dp | 2.3.s5 | dangling pointer | | |
| Apache | 2.0.47 | stack overflow | 283K | a Web server |
| CVS | 1.11.4 | double free | 114K | a version control server |

19

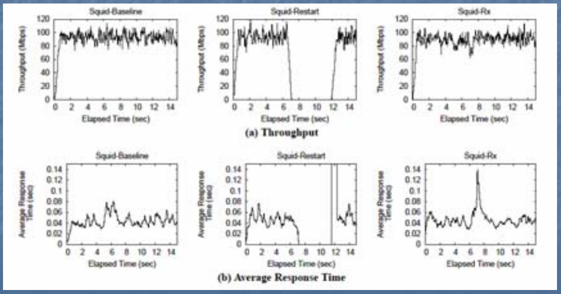
## Overall Results



| Apps | Bugs | Failure Symptoms | Environmental Changes | Clients Experience Failure? | | Recoverable? | | Average Recovery Time (s) | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Alternatives | Rx | Alternatives | Rx | Restart | Rx |
| Squid | Buffer Overflow | SEGV | Padding | Yes | No | No | Yes | 5.113 | 0.095 |
| MySQL | Data Race | SEGV | Schedule Change | Yes | No | 40% probablity | Yes* | 3.500 | 0.161 |
| Apache | Stack Overflow | Assert | Drop User Request | Yes | No | No | Yes | 1.115 | 0.026 |
| CVS | Double Free | SEGV | Delay Free | Yes | No | No | Yes | 0.010 | 0.017 |
| Squid-ui | Uninit Read | SEGV | Zero All | Yes | No | No | Yes | 5.000 | 0.126 |
| Squid-dp | Dangling Pointer | SEGV | Delay Free | Yes | No | No | Yes | 5.006 | 0.113 |

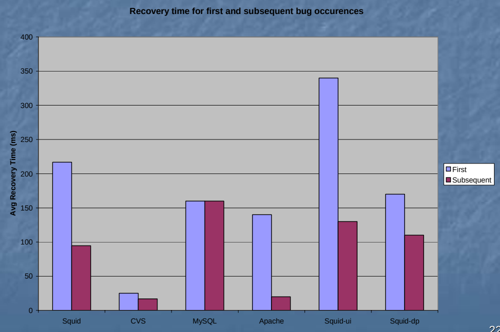
20

## Throughput and Avg Response Time



21

## Recovery Time



22

## Rx Advantages

- Comprehensive
  - Can survive many common software defects
- Safe
  - Does not change program, only environment it runs in
- Noninvasive
  - Few to no modifications required in software (no mods in any of the tested systems)
- Efficient
  - No rebooting (mostly) with little overhead
  - Learns from previous solutions
- Informative
  - Bugs are shown and details are given on the nature of the bug

23

## Issues

- Unavoidable Bug/Failures
  - Accumulative memory leaks cannot be detected by Rx
  - Only solution is program restart
- Worst case scenario 2x time for normal restart
  - Did not happen in any of the tests

24

## Questions/Complaints?

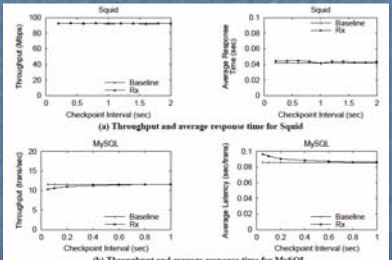
25

## What do they mean with “execution environment”?

- “almost everything that is external to the target application but can affect the execution of the target application”
- 3 levels:
  - Lowest: Hardware (processor, devices)
  - Middle: OS kernel (scheduling, virtual memory management, device drivers)
  - Highest: libraries (standard, third-party)

26

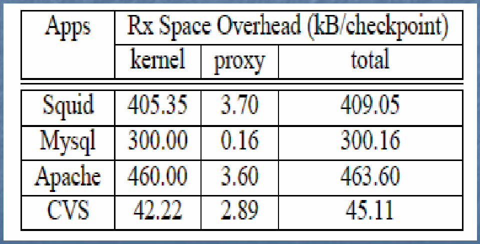
## Throughput and Avg Response Time



(a) Throughput and average response time for Squid
(b) Throughput and average response time for MySQL

27

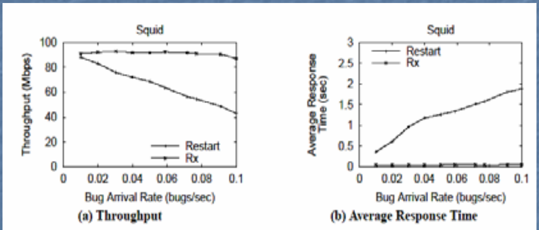
## Avg Space Overhead per Checkpoint



| Apps | Rx Space Overhead (kB/checkpoint) | | |
|---|---|---|---|
| | kernel | proxy | total |
| Squid | 405.35 | 3.70 | 409.05 |
| Mysql | 300.00 | 0.16 | 300.16 |
| Apache | 460.00 | 3.60 | 463.60 |
| CVS | 42.22 | 2.89 | 45.11 |

28

## Different bug arrival rates



(a) Throughput
(b) Average Response Time

Figure 5: Throughput and average response time with different bug arrival rates

29